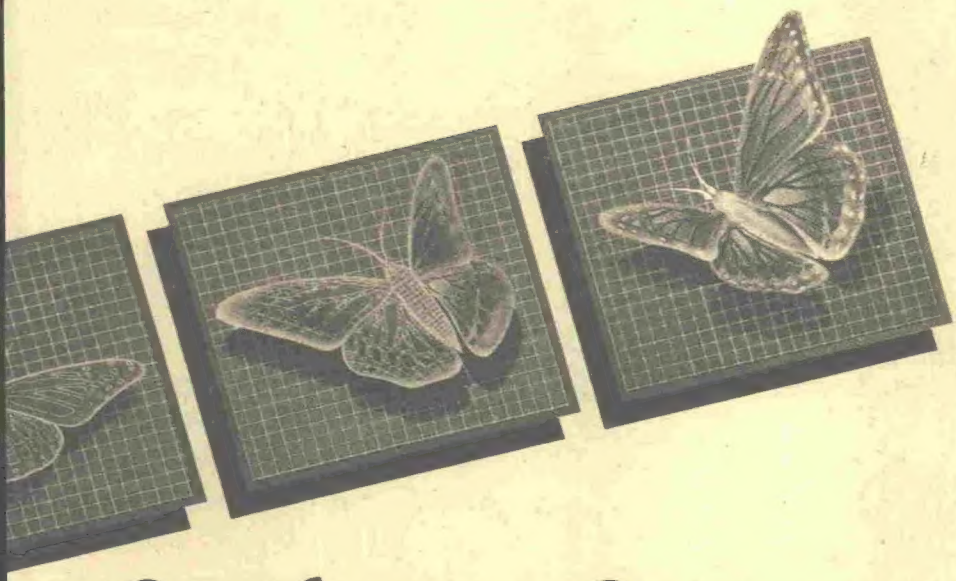


พิมพ์รูปกราฟิกจากจอเฮอร์คิวลิสบนเครื่องพิมพ์ดอตแมตริกซ์

เทคนิคการเขียนโปรแกรมพิมพ์รูปจากจอภาพผ่านทางการ์ดเฮอร์คิวลิส ลงบนเครื่องพิมพ์ดอตแมตริกซ์เขาทำได้อย่างไร

จเร เลิศสุดวิชัย

เคยใช้โปรแกรม GRAPHICS.COM ของดอส ช่วยพิมพ์รูปกราฟิกบนเครื่องพิมพ์ดอตแมตริกซ์ได้ แต่บนจอเฮอร์คิวลิสยังไม่เคยพบโปรแกรมช่วยเหมือน GRAPHICS.COM เลย เพราะว่าผู้เขียนต้องทำบทความส่งวารสารไมโคร ซึ่งต้องใช้รูปบนจอภาพประกอบด้วย จึงจำเป็นต้องเขียนโปรแกรมช่วยเองด้วย ก็เลยนำลงมาเป็นของฝากให้ผู้อ่านใช้งานบ้าง โปรแกรมที่นำลงเป็นโปรแกรมเรสซิเดนต์ที่เขียนด้วยภาษาซีใช้กับเครื่องพิมพ์ EPSON

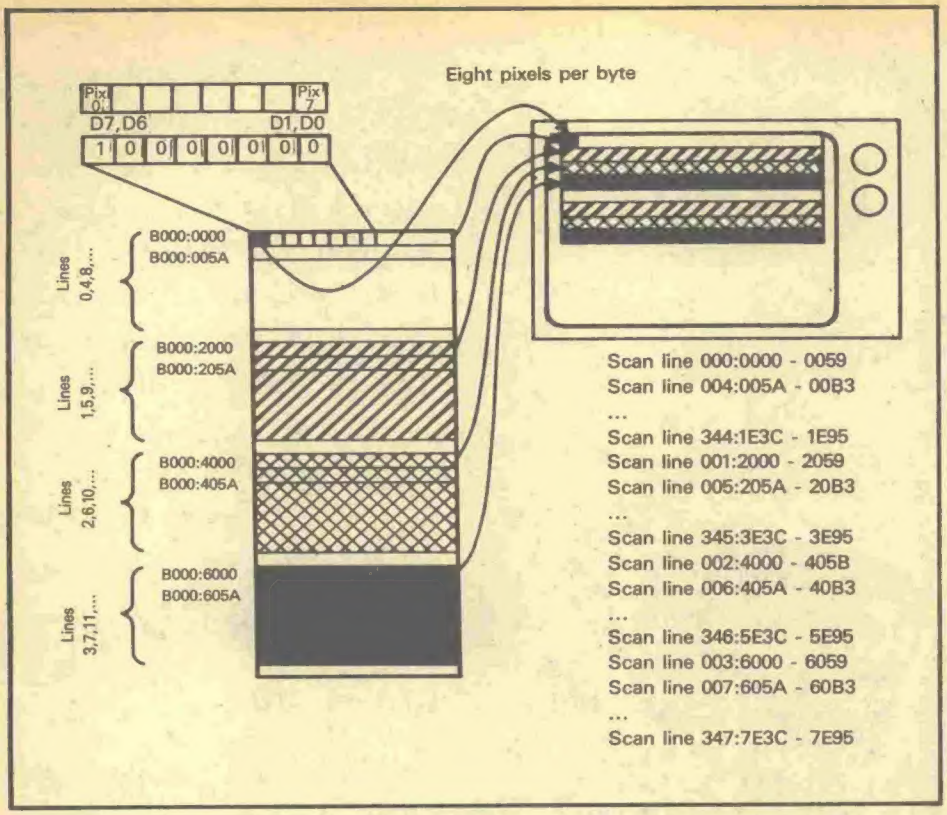
ข้อมูลกราฟิกบนจอเฮอร์คิวลิส

จอภาพเฮอร์คิวลิสในโหมดกราฟิกจะมีความละเอียด 720 จุดในแนวนอน และ 348 จุดในแนวตั้ง จุดหนึ่งบนจอภาพจะถูกแทนข้อมูลขนาด 1 บิต ตำแหน่งหรือลำดับของจุดบนจอภาพจะเรียงต่อเนื่องกัน 8 จุดจึงแทนได้ด้วยข้อมูล 1 ไบต์ แต่ว่าตำแหน่งของหน่วยความจำที่แทนจุดจะไม่มีเรียงต่อกัน ดังรูปที่ 1 และหน่วยความจำบนการ์ดควบคุมจอภาพ สามารถแทนภาพได้ถึง 2 เพจ คือเพจที่ 0 เริ่มหน่วยความจำเซกเมนต์ที่ B000h และเพจที่ 1 จะเริ่มต้นหน่วยความจำเซกเมนต์ที่ B800h ส่วนออฟเซตสามารถหาได้ถ้าทราบตำแหน่งของจุดเป็น X, Y โดยใช้สมการนี้

ตำแหน่งของหน่วยความจำ = ((Y &3) << 13) + 90* (Y >> 2) + (X >> 3)

ควบคุมเครื่องพิมพ์พิมพ์กราฟิก

การควบคุมเครื่องพิมพ์ให้พิมพ์ในโหมดของกราฟิกจะต่างกับการพิมพ์ในโหมดข้อความทั่วไป ในโหมดข้อความข้อมูล 1 ไบต์จะทำให้เครื่องพิมพ์



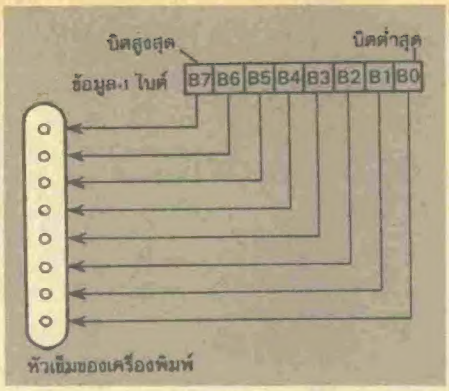
รูปที่ 1 แสดงความสัมพันธ์ระหว่างจุดบนจอภาพเฮอร์คิวลิสกับหน่วยความจำ

พิมพ์ตัวอักษรออกมา 1 ตัว แต่ในโหมดกราฟิกข้อมูล 1 ไบต์(ในกรณีเครื่องพิมพ์หัวเข็มขนาด 24 เข็มจะใช้ 3 ไบต์) จะเป็นการสั่งให้เครื่องพิมพ์พิมพ์ข้อมูลออกมาเป็นจุด แต่ว่าจุดจะเรียงกันในแนวตั้ง ตัวอย่างดังในรูปที่ 2 ซึ่งแสดงความสัมพันธ์ระหว่างข้อมูล 1 ไบต์กับจุด 8 จุดที่จะถูกพิมพ์ออกมาโดยเครื่องพิมพ์

ถ้าเป็นกรณีเครื่องพิมพ์ 24 เข็มก็จะเป็นลักษณะเดียวกันคือใช้ข้อมูล 3 ไบต์เรียงต่อกันเพื่อควบคุมหัวเข็มในแนวตั้ง 24 เข็ม

เลือกโหมดความหนาแน่นกราฟิกของเครื่องพิมพ์

ก่อนที่จะส่งข้อมูลให้เครื่องพิมพ์เพื่อควบคุมหัวเข็มของเครื่องพิมพ์ จะต้องส่งคำสั่งเลือกโหมดความหนาแน่นของจุดในการพิมพ์กราฟิกก่อน ซึ่งคำสั่งในการเลือกโหมดความหนาแน่นจะประกอบด้วย รหัสควบคุมดังนี้



รูปที่ 2 แสดงความสัมพันธ์ระหว่างข้อมูลที่ควบคุมการพิมพ์ของหัวเข็ม

CHR$(27) + "*" + CHR$(m) + CHR$(n1) + CHR$(n2) + CHR$(d1) + ...CHR$(dn)

โดย m จะเป็นตัวเลือกความหนาแน่นของจุดดังในตารางรูปที่ 3 ส่วน n1 และ n2 จะบอกถึงความยาวของจุดในแนวนอน ซึ่งจะหาได้จาก n2×256+n1 ซึ่งในทำนองกลับกัน ถ้าทราบความยาวของจุดในแนวนอน เช่น มี N จุดก็จะสามารถหาค่า n1 และ n2 ได้ดังนี้

n1 = N%256
n2 = N%256

	m	จำนวนเข็ม	m	จำนวนเข็ม	ความหนาแน่นในแนวนอน dots/inch
Normal-density	0	8	32	24	60
Double-density	1	8	33	24	120
Double-speed	2	8			120
Double-density					
Quadruple-density	3	8			240
CRT-graphics	4	8			80
CRT-graphics II	6	8	38	24	90
Triple-density			39	24	180

รูปที่ 3 ตารางแสดงโหมดความหนาแน่นของจุดในการพิมพ์รูปกราฟิก

ในกรณีเลือกหัวพิมพ์ 8 เข็มคือ m มีค่าเป็น 0 ถึง 4 และ 6 จำนวนข้อมูลที่ต้องส่งตามมาก็จะเท่ากับจำนวนความยาวของจุดในแนวนอน แต่ถ้าเป็นกรณีเครื่องพิมพ์ 24 เข็ม คือเลือก m มีค่าเป็น 32, 33, 38 หรือ 39 จำนวนข้อมูลที่ต้องส่งจะมีจำนวน $(n2 \times 256 + n1) \times 3$ หรือเป็น 3 เท่าของกรณีหัวพิมพ์ 8 เข็ม จำนวนข้อมูลที่ส่งเกินเครื่องพิมพ์จะไม่สนใจ

ทดสอบการพิมพ์แบบกราฟิก

คราวนี้ลองทดลองสั่งพิมพ์ในโหมดกราฟิกดูบ้าง โดยข้อมูลที่จะควบคุมหัวเข็มจะเป็นดังรูปที่ 3 โดยจะทำการส่งข้อมูลทั้งสองสลับกันไปมาเป็นจำนวน 300 จุด

โปรแกรมสำหรับควบคุมการพิมพ์อยู่ในรูปที่ 4 และแสดงผลลัพธ์ที่ได้อยู่ท้ายโปรแกรม

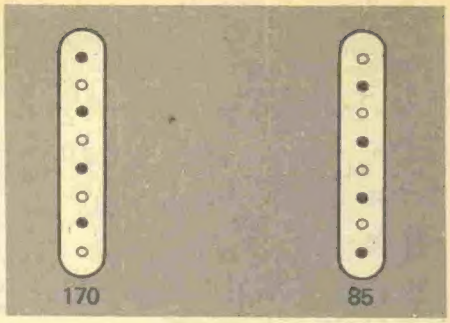
จัดระยะห่างในแนวตั้ง

ในการพิมพ์แบบกราฟิกจะต้องทำการจัดระยะห่างระหว่างบรรทัดใหม่เพื่อให้จุดในแต่ละบรรทัดชิดต่อเนื่องกัน โดยการส่งคำสั่ง

CHR$(27) + "3" + CHR$(n)

โปรแกรมในรูปที่ 5 จะเป็นการพิมพ์กราฟิกทั้งหมด 6 บรรทัด โดยทำการปรับระยะห่างระหว่างบรรทัดแล้วซึ่งจะได้จุดห่างระหว่างบรรทัดต่อเนื่องกัน สังเกตว่าจบแต่ละบรรทัดจะต้องส่งรหัสขึ้นบรรทัดใหม่ตามไปด้วย

รูปที่ 3 ข้อมูลสำหรับควบคุมหัวเข็มเครื่องพิมพ์ ซึ่งจะทำให้หัวเข็มพิมพ์พิมพ์สลับกัน



สั่งพิมพ์ทิศทางเดียว

ตามปกติเครื่องพิมพ์จะพิมพ์ในลักษณะ 2 ทิศทาง ซึ่งในลักษณะของ

```
#include <stdio.h>

#define  N 300

main ()
{
        int n1,n2,j;
        FILE *fp;

        fp = fopen ("LPT1","wb");
        n1 = N % 256;
        n2 = N / 256;
        fprintf (fp,"%c%c%c%c%c",27,'*',6,n1,n2);
        for (j=0; j<N/2 ; j++)
                {
                putc (170,fp);
                putc (85,fp);
                }
        putc ('\n',fp);
        fclose (fp);
}
```

รูปที่ 4 โปรแกรมตัวอย่างพิมพ์กราฟิกโหมดและผลลัพธ์ของโปรแกรม

```
#include <stdio.h>

#define  N 400

main ()
{
        int n1,n2,i,j;
        int oddline,oddquadruple;
        FILE *fp;

        fp = fopen ("LPT1","wb");
        n1 = N % 256;
        n2 = N / 256;
        fprintf (fp,"%c%c%c",27,'3',24);
        for (i=0; i<6; i++)
                {
                oddline = i % 2;
                fprintf (fp,"%c%c%c%c%c",27,'*',6,n1,n2);
                for (j=0; j<N ; j++)
                        {
                        oddquadruple = j% 8 > 3;
                        putc (oddquadruple == oddline ? 170 : 85,fp);
                        }
                putc ('\n',fp);
                }
        fclose (fp);
}
```

รูปที่ 5 โปรแกรมตัวอย่างพิมพ์กราฟิกหลาย ๆ บรรทัดต่อกันและผลลัพธ์ของโปรแกรม

การพิมพ์กราฟิกจะทำให้จุดห่างระหว่างบรรทัดที่อยู่ต่อกันจะไม่ตรงกัน ยิ่งถ้าเป็นเครื่องพิมพ์ที่ถูกใช้งานมานานจะเยื้องกันมากขึ้น แต่สามารถจะแก้ไขได้โดยสั่งให้เครื่องพิมพ์พิมพ์ในทิศทางเดียวเท่านั้นด้วยคำสั่ง

CHR$(27) + "u" + CHR$(n)
โดยที่
n = 1 สั่งพิมพ์ในทิศทางเดียว
n = 0 สั่งพิมพ์ทั้งสองทิศทาง

สั่งพิมพ์ผ่านไบออส

เนื่องจากโปรแกรมที่ใช้งานต่อไปเป็นโปรแกรมเรสซิเดนต์แบบง่าย ๆ ซึ่งไม่ได้ตรวจสอบสภาวะความปลอดภัยในการทำงานของโปรแกรมเรสซิเดนต์ ดังนั้นการส่งข้อมูลออกทางเครื่องพิมพ์จึงจำเป็นต้องส่งผ่านทางไบออสโดยอาศัยฟังก์ชันในรูปที่ 6

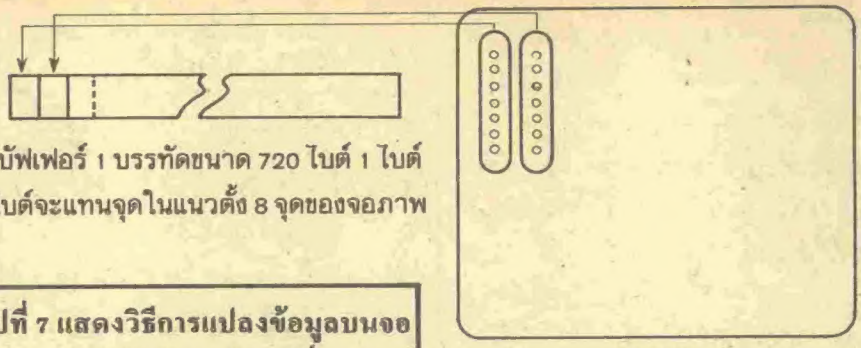
```
/**********************************
 * Printer Character to printer   *
 * use BIOS service int 17h.      *
 * Return Status of printer.      *
 **********************************/
void PrnChar (int c)
{
        union REGS regs;

        if (PrinterReady)
                {
                regs.x.dx = 0;
                regs.h.ah = 0;
                regs.h.al = c;
                int86 (0x17,&regs,&regs);
                if ((regs.h.ah & 0x01) && PrinterReady)
                        PrinterReady--;
                }
}
```

รูปที่ 6 ฟังก์ชันการส่งข้อมูลพิมพ์ผ่านทางไบออส

จัดข้อมูลบนจอภาพกราฟิกให้อยู่ในรูปแบบที่ส่งพิมพ์ได้

จะเห็นลักษณะข้อมูลบนจอภาพซึ่งถูกจัดเก็บในหน่วยความจำเรียงต่อกันในแนวนอน แต่ละลักษณะของข้อมูลที่เราจะส่งพิมพ์จะต้องส่งพิมพ์ในแนวตั้ง 8 จุดด้วยกัน ดังนั้นจะต้องทำการแปลงข้อมูลบนจอภาพให้อยู่ในรูปแบบที่สามารถส่งพิมพ์ได้ การส่งพิมพ์นั้นจะส่งพิมพ์ทีละ 1 บรรทัด จึงต้องจองหน่วยความจำไว้ให้พอกับข้อมูลหนึ่งบรรทัดที่จะส่งพิมพ์ แล้วคำนวณหาจุดบนจอภาพจากนั้นจัดเก็บลงในหน่วยความจำที่จองเอาไว้ให้ตรงกับลักษณะที่จะพิมพ์



รูปที่ 7 แสดงวิธีการแปลงข้อมูลบนจอภาพให้อยู่ในรูปของเครื่องพิมพ์

การใช้งานโปรแกรม

ตัวอย่างโปรแกรมที่ให้ไว้ในรูปที่ 8 เป็นโปรแกรมเทอร์โบซี ซึ่งต้องคอมพายล์ในแบบคอมมานด์ไลน์ คือคอมพายล์ด้วย

TCC-ml-B Hercdump .C

สำหรับผู้อ่านที่ไม่ต้องการพิมพ์โปรแกรมเองในการใช้งานโปรแกรมก็หาได้จากซอฟต์เมล์ฉบับนี้ครับ เนื่องจากภาพกราฟิกโหมดบนเฮอร์คิวลิสมีอยู่ 2 เพจ และการทำงานในแต่ละโหมดไม่สามารถจะตรวจสอบได้ด้วยโปรแกรม ดังนั้นผู้ใช้ต้องทำการติดตั้งเอง โดยใส่หมายเลขเพจต่อท้ายชื่อโปรแกรม เช่น

A> HercDump 0

ตัวโปรแกรมจะตรวจสอบตัวเองว่าเคยโหลดมาแล้วหรือยัง ถ้าเคยโหลดมาแล้วก็จะไม่โหลดซ้ำอีกแต่จะเปลี่ยนเพจที่พิมพ์ให้ตามหมายเลขเพจที่อยู่ท้ายชื่อโปรแกรม และเมื่อต้องการพิมพ์รูปภาพออกเครื่องพิมพ์ก็กดคีย์ ctrl และ shift ขวาพร้อม ๆ กัน

รูปที่ 8 โปรแกรมพิมพ์กราฟิกจากจอเฮอร์คิวลิสลงเครื่องพิมพ์

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dos.h>

#define FullFeed        24
#define Xmax            720
#define Ymax            348
/* Hot Key = Ctrl + Right Shift */
#define MarkKey         0x05

extern unsigned _stklen  = 64;
extern unsigned _heaplen = 0;
unsigned sizeprogram;
unsigned myss;
unsigned stack;
```

→

```
/* Old interrupt KBD 9 */
void interrupt far (* oldintr9) ();

/* Bitwise for test bit in byte data */
static int bits [8] = { 128,64,32,16,8,4,2,1 };

/* A row buffer bit image for FX or LQ printer */
char PrintBuffer [720];

/* Hot Key to Print Graphics screen */
int far *HotKey = (int far *)MK_FP (0x0040,0x0017);
int PrinterReady = 1;
char far *PagePtr;
char far *ScrnPtr;

int Pixlist (int Xcor,int Ycor)
{
        unsigned int GraphOfs;

        GraphOfs = ((Ycor & 3) << 13) + 90*(Ycor >> 2) + (Xcor >> 3);
        if (*PagePtr==0)
                ScrnPtr = (char far *)MK_FP (0xb000,GraphOfs);
        else
                ScrnPtr = (char far *)MK_FP (0XB800,GraphOfs);
        return (*ScrnPtr & bits [Xcor % 8]);
}

/****************************************
 * Printer Character to printer         *
 * use BIOS service int 17h.            *
 * Return Status of printer.            *
 ****************************************/
void PrnChar (int c)
{
        union REGS regs;

        if (PrinterReady)
                {
                regs.x.dx = 0;
                regs.h.ah = 0;
                regs.h.al = c;
                int86 (0x17,&regs,&regs);
                if ((regs.h.ah & 0x01) && PrinterReady)
                        PrinterReady--;
                }
}

/******************
 * Reset printer  *
 ******************/
void ResetPrinter ()
{
        PrnChar (27);
        PrnChar ('@');
}

/************************************
 *  Set printer to graphic mode print *
 ************************************/
void SetPrinter ()
{
        int n1,n2;

        n1 = Xmax % 256;
        n2 = Xmax / 256;
        PrnChar (27);PrnChar ('3');PrnChar (24);
        PrnChar (27);PrnChar ('*');PrnChar (6);
        PrnChar (n1);PrnChar (n2);
}

/*********************************************
 * Convert screen data to printer data format *
 *********************************************/
void ConvertOneRow (int row,int line)
{
        int Xcor,Ycor,EndLine,StartLine;

        StartLine = row*8;
        EndLine = StartLine + line;
        memset (PrintBuffer,0,Xmax);
        for (Xcor=0;Xcor<Xmax;Xcor++)
                for (Ycor=StartLine;Ycor<EndLine;Ycor++)
                        PrintBuffer [Xcor] |=
                        (Pixlist (Xcor,Ycor) ? bits [Ycor-StartLine] : 0);
}

void PrnDataOneRow ()
{
        int column;

        for (column=0;column<Xmax;column++)
                PrnChar (PrintBuffer [column]);
}

void PrnOneRow (int row,int line)
{

        ConvertOneRow (row,line);
/*      Reverse ();         */
        SetPrinter ();
```

→

รูปที่ 8 (ต่อ)

```
        PrnDataOneRow ();
        PrnChar ('\n');
}

/*************************************
 *   Print Screen interrupt function *
 *************************************/
void DumpScr ()
{
        int i;

        if (!PrinterReady)
                PrinterReady++;
        ResetPrinter ();
        SetPrnUniDirection ();
        for (i=0;i<43;i++)
                PrnOneRow (i,8);
        PrnOneRow (43,4);
        ResetPrinter ();
}

/*************************************
 *   Dummy function hold data for    *
 *   interrupt Print screen function *
 *************************************/
void ControlDumpScrn ()
{
        asm _Signature   db  'DumpScr Herc',0
        asm _PageDump    db  0
}

void interrupt far intr9 ()
{
        if ((*HotKey & MarkKey) == MarkKey)
                DumpScr ();
        (* oldintr9) ();
}

/*************************************************
 *  Install interrupt vector and stay resident *
 *************************************************/
void resident ()
{
        struct SREGS seg;

        segread (&seg);
        myss = seg.ss;
        stack = _SP;
        setvect(0x9, intr9);
        sizeprogram = (unsigned)myss +
                (((unsigned)stack + 50) / 16) -
                (unsigned) _psp;
        keep (0,sizeprogram);
}

char far *StringTest;
main (int argc,char *argv[])
{
        int Page;

        oldintr9 = getvect (0x9);
        StringTest = (char far *)oldintr9;
        StringTest -= 15;
        if (strcmp (StringTest ,"DumpScr Herc") != 0)
                {
                if (argc > 1)
                        {
                        Page = atoi (argv[1]);
                        PagePtr = (char far*) intr9;
                        PagePtr--;
                        *PagePtr = Page;
                        }
                puts ("Print Screen Herc, Now install");
                resident ();
                }
        else
                {
                puts ("Print Screen Herc, Already install");
                if (argc > 1)
                        {
                        PagePtr = StringTest + 14;
                        Page = atoi (argv[1]);
                        *PagePtr = Page;
                        }
                }
}
```